Design and Simulation Analysis of a Time Predictable Computer Architecture

K. M. Shivaraj, * P. Padma Priya Dharishini

Faculty of Engineering and Technology, M. S. Ramaiah University of Applied Sciences, Bangalore 560 054

*Contact Author e-mail: padmapriya.cs.et@msruas.ac.in

Abstract

General purpose processors are driven by the idea of optimizing for average case execution time analysis. But the processors used in real-time embedded applications must undergo timing analysis to ensure the timing constraints are met or not. Hence the Worst-Case Execution Time (WCET) analysis is important for real time applications. A Time Predictable Computer Architecture helps in simplifying the WCET analysis. In this Paper, a Time Predictable Computer Architecture is developed. This architecture is characterised by a four stage pipeline, in order execution, basic RISC instruction set, timing instructions and scratchpad memory. The four stage pipeline is selected as a trade-off between real time performance and time predictability. It is designed to detect and handle hazards by setting and resetting register when the registers are accessed to write or read. An on-chip scratchpad memory is used as an alternative to cache memory. All the instruction are loaded from the instruction memory to scratchpad memory before execution. The instructions are only executed in-order, and timing instructions are implemented to specify and ensure deadline constraints. These features make the developed architecture time predictable in nature. The developed architecture is simulated and analysed using the HASE (Hierarchical Architecture Simulation Environment) simulator. A layout of the architecture is developed to visualise the instruction flow. Simulation results show that the developed computer architecture remains time predictable when tested with varying sets of simple instructions. It would be interesting to incorporate features such as cache, branch prediction and out-of order execution into the developed architecture and analyse its effect on time predictability.

Key Words: Time Predictable Computer Architecture, Instruction Set, Pipeline, WCET, Scratchpad Memory

1. INTRODUCTION

An embedded system plays an important role in modern society and they are frequently used in many real-time applications which include household appliances, automotive industry, avionics, etc. A real-time application system has to produce logically correct results within a specified period of time. If a correct computation result is delayed, it will be considered useless. In case of a hard real-time system, a late computation may cause a critical system failure leading to disastrous consequences, possibly endangering human life. A soft real-time system can tolerate results that occasionally miss their deadlines. The system still produces correct results, but accompanied by an inferior service quality. Such applications must undergo timing analysis to ensure the timing constraints are met and the mission succeeds. Worst-Case Execution Time (WCET) is the maximum execution time of a program on any processor for any given input. WCET yields safe and precise upper bounds of tasks for a given hardware platform [1]. The measurement based analysis methods are preferred because they guarantee to consider all possible execution times. Time predictable characteristics of computer architecture help in analyzing the performance of the real time applications.

Time predictability is one of the most important design considerations for real-time systems. It is observed that many features in standard architecture such as pipelining, cache memories and branch prediction are in favor of average case performance that significantly compromises the time predictability. WCET analysis is quite complex on modern computer systems since modern processors contain features like pipelines or caches and maintains an internal state to improve peak performance. The development of more predictable architecture will reduce the complexity of WCET analysis.

Therefore, the time predictability of computer architecture and its compiler support is studied. The impediments to time predictability for computer architecture are analyzed and architecture-based techniques to address these problems with minimal modifications to the architecture design are proposed. Specifically, the computer architecture is designed to enhance the WCET analysis for computer architecture.

The WCET of a program is the maximum execution time it can take on concrete target hardware. The knowledge of the WCET of the tasks is crucial for the designing of real-time systems. It becomes possible to verify the timeliness of the whole real-time system, only if the safe upper bounds for the WCET of all time-critical tasks have been established.

The motivation of this work is to design computer architecture with time predictable characteristics for real time applications and implement the same using the HASE simulator [2] and analyze its performance.

2. PROBLEM DEFINITION

Time predictability issues arise because of the cache memory and pipeline. The architecture proposed in this paper deals with the design and simulation of a computer architecture with time predictable characteristics for real time applications by considering cache memory and pipelining. Time predictable computer architecture makes the WCET analysis easier.

3. SYSTEM REQUIREMENTS

Based on the stated problem definition, the requirements of the architecture can be formally stated as:

- FNR1: The developed architecture should be Time Predictable in nature
- FNR2: The developed architecture should be able to accurately execute basic ALU instructions and Timing instructions
- FNR3: The developed architecture should be able detect and handle hazards in pipeline stages

4. ARCHITECTURE OVERVIEW

The 32-bit RISC architecture is designed which is based on a simplified version of the standard architecture. The architecture has 32-bit instruction words and 32 general purpose registers. An external clock is used as the timing mechanism for the control and data path units. This section includes a description of the main components and features of the architecture and the instruction word formats.

The basic data path of a architecture is shown in Fig. 1. The Instruction Fetch loads the instruction pointed by the program counter (PC) from memory. The Instruction Decoder then generates the appropriate control signals for the Execute unit, which performs the desired function (arithmetic, logic, etc.) on the data. The Memory access and Write back unit then updates the memory and registers respectively with any new values.

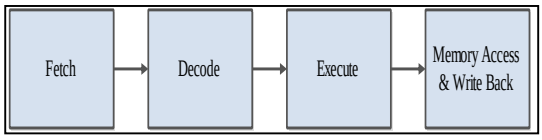

Fig. 1 Architecture Data Path

The architecture is designed with load/store and Von Neumann architecture model. One shared memory for instructions and data with one data bus and one address bus between processor and memory. Instruction and data are fetched in sequential order so that the latency incurred between the machine cycles can be reduced. Four stages of pipelining have been incorporated in the design which increases the speed of operation. The pipelining stages are fetch, decode, execute and memory access/write back.

4.1 Flow Chart

The Program Counter (PC) gets a memory address of a program and that PC value is used as an index to the instruction memory which supplies a next instruction to be executed. The instruction is retrieved from main memory and the PC is incremented (PC=PC+1). The fetched instruction is fed to the decode stage where the instruction is decoded by accessing register file and it loads operand values from register file. Then the decoded instruction is fed to execute stage i.e. ALU, where the instructions are processed and ALU operation is performed and the results are obtained. The obtained results are stored back to register file or to data memory.

For Immediate type instructions, data is written back into register File. For Load operation, memory is accessed for loading data to register. Store operation, writes data into memory from register file and new program counter is updated for Jump/Branch instructions. This flow is shown in Fig. 2.

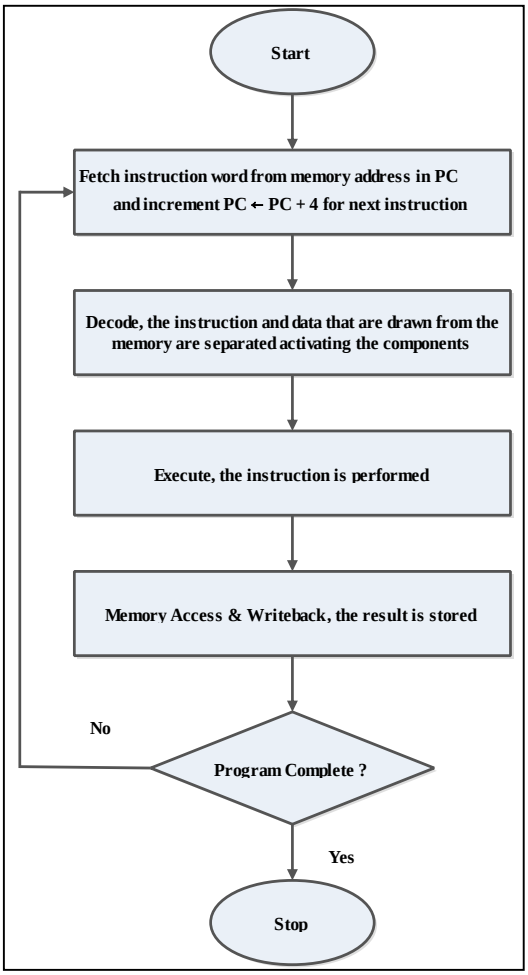

Fig. 2 Flowchart of Time Predictable Computer Architecture

4.2 Pipeline

The complete architecture is divided into 4 distinct activities called 4 stages pipelining, which is shown in the Fig. 3. The four stages in the designed pipeline architecture are:

1. Instruction fetch (IF)
2. Instruction Decode (ID)
3. Execution (EX)
4. Memory Read/Writeback (MW)

4.3 Instruction Set

The instruction set for the designed architecture is divided into four categories.

a. Arithmetic and Logic Instructions
b. Memory Instructions
c. Branch Instructions
d. Timing Instructions [3]

Timing instructions are included to control the timing of a processor. It allows the programmer to set cycle-accurate time. The deadline of the instructions is specified explicitly in terms of the number of clock cycles. The instruction should finish its execution before the specified deadline. If the provided deadline runs out before the enclosed instruction has finished, it will throw an exception and stops simulation.

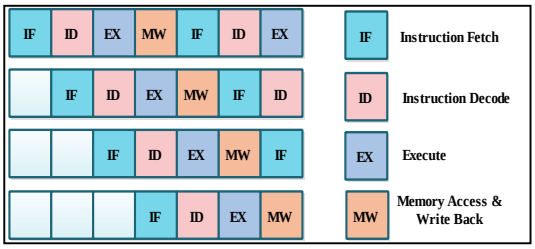

Fig. 3 Four Stage Pipeline

4.4 Memory Organization

The Memory contains two arrays, one for instructions and one for data. These are held separately because instructions are held in readable (string) form for visualization purposes whilst data values are held as integers. Each word in both memories contains its own byte address as well as its instruction or data. Each word in the Main Registers array contains an index number field, a data field and a 'Busy Bit' field.

The scratchpad memory is designed instead of cache memory. Scratchpad memory is an on-chip memory which copies the instructions from the instruction memory before executing [4]. Hence the instructions are accessed from scratchpad memory instead of instruction memory. The size of the scratch pad memory depends on the application.

5. RESULTS AND DISCUSSION

The implemented architecture is tested for its functionality. The obtained results are discussed in this section. Fig. 4 shows the architecture layout, designed and implemented in HASE simulator [5].

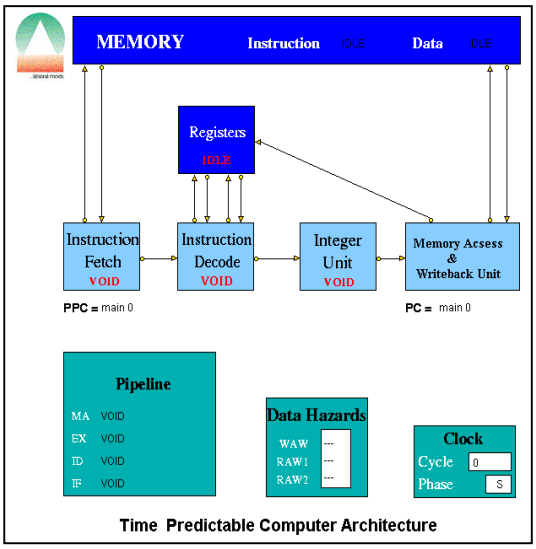

Fig. 4 Architecture Layout

In this paper we had taken assembly code to show time predictability. Fig. 5 shows the basic register based instructions that are executed in the designed architecture.

instr_mem

Index	Value
	ADD R1 R0 R1
	SUB R4 R3 R2
	MUL R2 R3 R5
	DIV R5 R6 R3
	NOP

Fig. 5 Arithmetic Instructions

Fig. 6 shows the execution of arithmetic instructions in the HASE simulator. Fig. 7 shows the timing.

Fig. 6 Arithmetic Instructions Flow in Pipeline instructions that are executed in the designed architecture

instr_mem

Index	Value
	ADDI R2 R0 15
	SUB R2 R2 R1
	DEAD R4 8
	DADD R1 R0 R1
	BREAK
	NOP

Fig. 7 Timing Instructions

Fig. 8 shows the results of the executed timing instruction.

```
Current Cycle no:10

Sum:17*****Correct::Executed *****
```

Fig. 8 Output of Timing Instruction

Fig. 9 shows the simple instructions which have arithmetic instructions, branch instructions and memory instructions which creates hazard in the pipeline.

instr_mem

Index	Value
	ADDI R2 R0 15
	SUB R2 R2 R1
	ADDI R1 R0 1
loop1	ADDI R1 R1 1
	ADD R3 R1 R0
loop2	ADD R3 R3 R1
	SLT R4 R2 R3
	BNE R4 R0 done
	SLL R5 R3 2
	J loop2
	SW R0 0(R5)
done	BNE R1 R2 loop1
	NOP
	BREAK

Fig. 9 Instruction Memory Containing Instructions

Fig. 10 shows the data hazard [6] in the pipeline. The occurrences of hazards are shown in data hazard

window. Here the data hazard is handled by using Use bits. The Use bit for a Register R2 is required as a source operand is set, so there is a RAW hazard, as shown in the Data Hazards Display window. The Use bit for a Register R2 is required as a destination operand is set, so there is a WAW hazard, as shown in the Data Hazards Display window.

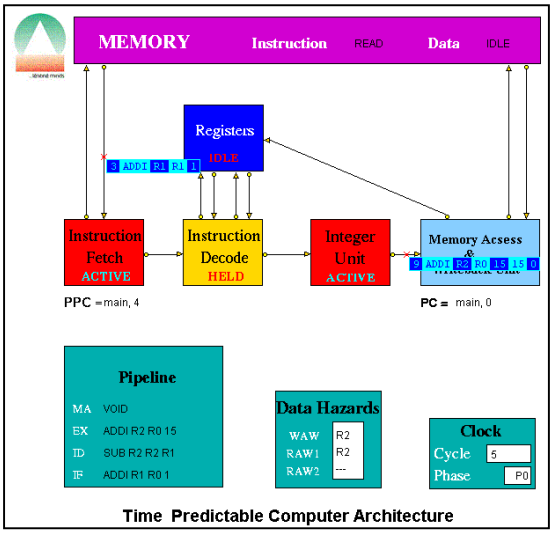

Fig. 10 Instructions Flow in Pipeline and Hazards in Pipeline

6. SUMMARY

The architecture designed in this paper is having timing instruction, scratch pad memory and 4-stage pipeline. These features make the developed architecture time predictable in nature. This architecture can be used for WCET analysis. The architecture is designed using HASE simulator.

This paper demonstrates how time predictable computer architecture is achieved by implementing and integrating the basic components of the standard architecture, a basic instructions set, pipelining, and scratchpad memory. The design of 4 stage pipelining, which is one of the primary concepts to speed up execution of instructions is emphasized in this paper. The use of scratchpad memory makes architecture time predictable, which is On-chip memory.

This paper explains about the designed time predictable computer architecture, functionality and its implementation. The implemented architecture is analyzed based on the results obtained. The developed architecture is suitable for small scale embedded systems.

7. CONCLUSIONS AND FUTURE WORK

- Use of in-order execution of instructions, replacement of cache with scratchpad memory and a 4 stage pipeline ensures time predictability of a basic RISC architecture
- The features such as cache memory, branch prediction and out-of-order instruction execution are not really desirable for time predictable architectures
- The architecture developed in this paper is not able to perform interrupt handling. This might be an interesting aspect to consider in the future, because interrupts are inherently unpredictable

REFERENCES

[1] Nilsen K.D., Rygg B., (1995) Worst-Case Execution Time Analysis on Modern Processors, *ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Real-Time Systems*, pp. 20-30.

[2] Howell F.W., Williams R., Ibbett, R.N., (1994) Hierarchical Architecture Design and Simulation Environment, *Proceedings of the Second International Workshop on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS, pp. 363-366.

[3] Lickly B., Liu I., Kim S., (2008) Predictable Programming on a Precision Timed Architecture, *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pp. 137-146.

[4] Banakar R., Steinke S., Lee B., (1999) Scratchpad Memory: A Design Alternative for Cache On-chip memory in Embedded Systems, *Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, pp. 73-78.

[5] Howell F.W., Ibbett R.N., (1998) Hierarchical Architecture Simulation Environment, *ACM Transactions on Modelling and Computer Simulation*, 8(4).

[6] Hennessy J., Patterson D., (2006) *Computer Architecture: A Quantitative Approach*, San Franscisco: Morgan Kaufmann.